



Rapport RIO 205

VUONG Christophe Sonia Bagumako
TELECOM Paristech

Table des matières

| | |
|-------------------------------------------------------|------------|
| Introduction | iii |
| 1 Docker | 1 |
| I Préliminaires | 1 |
| II Construction de l'image conteneur docker | 1 |
| III Travail dans le conteneur | 2 |
| IV Apache2 | 2 |
| V Dockerfile | 2 |
| VI Bilan | 3 |
| 2 VirtualBox | 4 |
| I Installation de VirtualBox | 4 |
| II Montage de l'image disque Ubuntu | 4 |
| III Configuration et réseau | 4 |
| IV Apache2 | 5 |
| V Analyse et comparaison avec Docker | 5 |
| 3 Vagrant | 6 |
| I Lancement de Vagrant | 6 |
| II Mise à jour du Vagrantfile | 6 |
| III Modification du Vagrantfile | 6 |
| IV Caractéristiques | 7 |
| V Conclusion | 8 |
| 4 qemu-kvm | 10 |
| I Préliminaires | 10 |
| II Création de l'image | 10 |
| III Création de la machine virtuelle | 10 |
| IV Analyse des configurations possibles | 10 |
| V Bonus | 12 |
| 5 Conclusions et perspectives | 13 |
| I Les leçons apprises | 13 |
| II Les frontières de la recherche | 13 |
| A Captures d'écran | 14 |
| I Docker | 14 |
| II VirtualBox | 15 |

Table des figures

| | | |
|-----|-----------------------------------------------|----|
| 1 | Cheminement SSH de base | iv |
| 1.1 | Cheminement SSH pour docker | 1 |
| 3.1 | Schéma VirtualBox + Vagrant | 9 |
| 4.1 | Schéma qemu-kvm translation de code | 11 |
| 4.2 | Schéma qemu-kvm paravirtualisation | 12 |
| A.1 | Procédure manuelle pour docker | 15 |
| A.2 | Etapas VirtualBox | 17 |

Introduction

Le rapport qui suit vise à étudier les différentes méthodes de virtualisation et leurs mise en place dans un cadre relativement simple.

Avec la digitalisation des entreprises pour accroître leur productivité, la naissance de nouveaux métiers liés à l'internet et l'internet des objets, la constante augmentation des données, l'optimisation des coûts de la gestion des infrastructures informatiques est un sujet important des entreprises. Une des solutions pour répondre à la réduction des coûts est le cloud computing et la virtualisation, principale technologie qui permet le cloud. Différentes techniques de virtualisation existent avec des performances plus ou moins optimales. Ce TP a pour objectif, dans un premier temps, de créer des machines virtuelles avec différentes techniques (Docker, VirtualBox, Vagrant, etc) et de nous intéresser ensuite aux aspects réseaux virtuels.

En effet, on a implémenté pour chaque méthode de virtualisation étudiée un serveur apache2 sur la machine virtuelle étudiée et on s'est intéressée à comment reproduire la procédure automatiquement. Enfin, on a analysé les performances de chacune des solutions et ce que cela implique pour des applications plus générales

Ce rapport est divisé en 3 parties et une annexes

La partie 1 traite de **docker**.

La partie 2 s'intéresse à la virtualisation avec **VirtualBox**.

La partie 3 étudie Vagrant utilisé avec **VirtualBox**.

La partie 4 aborde notre travail sur **qemu-kvm**.

L'annexe détaille les étapes qui nous ont mené au rapport.

Dans ce qui va suivre, la machine virtuelle sur laquelle nous allons travailler se trouve dans une session d'un ordinateur (vnet15) se trouvant physiquement en salle C132. Etant donné que la séance de TP ne se déroule pas dans cette salle, une connexion sécurisée à distance est réalisée par (mot de passe : oltatv !)

```
ssh -X tpvigie@vnet15.r2.enst.fr
```

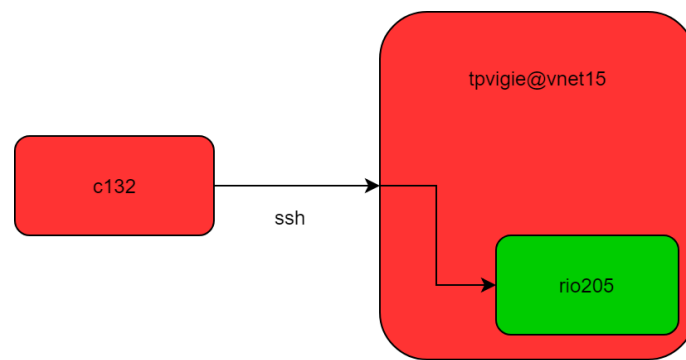
L'idée principale est de travailler du compte rio205 qui contient tous les éléments pour travailler avec des machines virtuelles. Or, ce compte est sur une machine virtuelle QEMU que l'on peut lancer avec le fichier de script bash via la commande :

```
tpvigie@vnet15:~$ cd Images
tpvigie@vnet15:~/Images$ ./lauchScript2.sh
```

On peut alors accéder au compte rio205 sur lequel on travaille via ssh pour des raisons pratiques. En ouvrant un nouvel onglet,

```
tpvigie@vnet15:~$ ssh -X -p 2222 rio205@127.0.0.1
```

Voici un schéma de l'environnement dans lequel on travaille.



Legende :

- Machine physique
- Machine virtuelle

FIGURE 1 – Cheminement SSH de base

Partie 1

Docker

I Préliminaires

Sur le terminal, on effectue d'abord l'installation de docker sur rio205.

```
rio205@rio205:~$ sudo apt-get update
rio205@rio205:~$ sudo apt-get install docker.io
```

Par commodité, pour lancer les commandes docker on a fait en sorte de permettre à l'utilisateur de lancer des commandes docker avec `sudo adduser rio205 docker`.

II Construction de l'image conteneur docker

On peut lancer un docker et donc créer un conteneur à l'aide de la commande suivante :

```
rio205@rio205:~$ docker run -it -p 8080:80 ubuntu
```

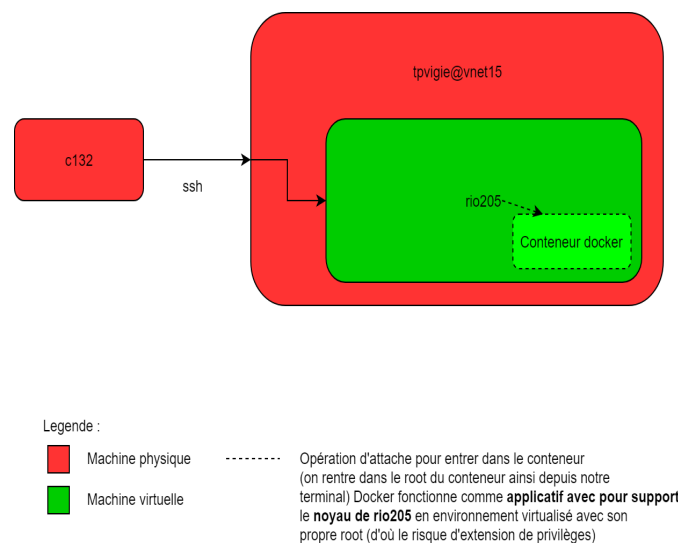


FIGURE 1.1 – Cheminement SSH pour docker

On a donc construit une image de container à l'aide cette commande en effectuant par la même occasion une redirection du port 80 au port 8080 de la localhost ici l'adresse ip 127.0.0.1. Elle s'appelle ubuntu, mais ne contient rien de particulier pour le moment. Elle a pour support la machine QEMU.

III Travail dans le conteneur

```
rio205@rio205:~$ sudo docker network inspect bridge
```

Cette commande permet de voir que la connexion du conteneur est de type bridge de l'adresse 172.17.0.2/16 du sous-réseau 172.17.0.0/16 à la localhost.

En effet, on a essayé avec plusieurs conteneurs ouverts pour voir ce qui se passait. La connexion semble être bridge par défaut et les adresses sont standards pour les conteneurs. Ils ont leur propre réseau qui n'interfère pas avec le réseau de la localhost. La connexion bridge apporte de la sécurité, car elle joue sur la couche logique, en permettant de filtrer les trames sur le réseau pour être transmis au conteneur. Un conteneur est donc isolé par ce procédé de bridge, car il ne peut pas intercepter les autres trames sur le réseau mais peut communiquer comme s'il faisait partie de ce réseau via rio205. Pour avoir accès au root du conteneur et agir dans celui-ci, il faut l'attacher par le procédé suivant :

```
rio205@rio205:~$ docker attach c1e1b9b3d548
```

C'est l'id donné par la commande `docker container ls`.

IV Apache2

Ainsi, on peut dans celui-ci installer ce qui faut pour lancer un serveur apache2 :

```
root@c1e1b9b3d548:/# apt-get update
root@c1e1b9b3d548:/# apt-get install apache2
root@c1e1b9b3d548:/# service apache2 status
```

On peut ensuite aller vérifier que l'on peut afficher une page Web (`index.html`). Le but est de pouvoir la personnaliser à souhait, en rajoutant par exemple nos noms.

On va donc dans le dossier par défaut qui stocke la page qu'affiche le serveur lorsque l'on ne tape dans un dossier précis du serveur.

```
root@c1e1b9b3d548:/# cd/var/www
```

Après avoir modifié le fichier `index.html`, il faut démarrer le service apache2 avec `start` ou `restart` :

```
root@c1e1b9b3d548:/# service apache2 start
```

On peut ensuite de l'extérieur du conteneur accéder aux données. Pour cela, on a fait attention de ne pas "sortir" du root. On a donc ouvert un nouvel onglet de terminal et recommencé tout le cheminement ssh pour arriver sur rio205 (cf annexe A pour les détails).

V Dockerfile

En dernier lieu, on a écrit un Dockerfile afin d'automatiser toutes les étapes précédentes, et même de lancer par exemple plusieurs conteneurs faisant tourner des services apache2.

```
rio205@rio205:~$ nano Dockerfile
```

```
# Here the dockerfile for apache2 task
FROM ubuntu:18.04
COPY myindex.html
RUN apt-get update \
    && apt-get install -y apache2
RUN mv myindex.html /var/www
RUN service apache2 restart
```

Listing 1: Dockerfile

On suppose que l'on a créé un fichier *myindex.html* qui contient le contenu d'une page Web (avec nos noms respectifs). On peut l'utiliser comme la page web que doit renvoyer le serveur Apache2 dont on va automatiser le démarrage de l'extérieur du conteneur.

Un des intérêts du Dockerfile est que l'on peut par exemple dupliquer le bout de code dans le fichier même pour créer 2 conteneurs. On n'est pas limité par le nombre de conteneurs dans un Dockerfile. C'est donc un puissant outil pour la mise à l'échelle de plusieurs tâches sur une machine physique en utilisant peu de ressources. En effet, celui-ci donne lieu à une image de conteneurs qui se base sur des couches ajoutées par les commandes COPY, RUN notamment. On se rend compte que si les instructions liées à celles-ci n'entraînent des changements indépendants, on n'a pas besoin à chaque fois que l'on met à jour *myindex.html* de mettre en place l'installation de apache2 dans l'image. En effet, la couche associée est déjà existante, et il suffira de la monter avec les autres couches modifiées avec `docker run`.

```
rio205@rio205:~$ docker build Dockerfile
rio205@rio205:~$ docker run -it -p 8080:80 ubuntu:18.04
```

Puis on peut vérifier que le serveur renvoie la page voulue, en lançant depuis rio205 :

```
rio205@rio205:~$ lynx http://127.0.0.1:8080/myindex.txt
```

VI Bilan

La technique Docker est intéressante à plusieurs points de vue. D'une part, un conteneur docker est très simple de création, ne demandant ni des connaissances sur la configuration d'OS, ni des installations de machines. D'autre part, la connexion bridge assure une certaine isolation des conteneurs vis-à-vis de la machine hôte (ici rio205). Enfin, on peut automatiser facilement la création de plusieurs images de conteneurs qui ont l'avantage de pouvoir être montées en plusieurs couches assurant une certaine granularité. Néanmoins, lorsque l'on stop un conteneur, il n'y a pas de mémoire par rapport aux tâches exécutées précédemment. Il faut donc prendre en compte ce facteur.

Partie 2

VirtualBox

I Installation de VirtualBox

Sur le terminal, on effectue d'abord l'installation de virtualBox sur rio205.

```
rio205@rio205:~$ sudo apt-get update
rio205@rio205:~$ sudo apt-get install virtualbox
```

Ici, il est important d'avoir écrit `ssh -X` pour que la sortie interface graphique soit elle aussi redirigée par connexion ssh. Le cheminement ssh qui en découle est plus compliqué que précédemment, car on parle bien d'une machine virtuelle qui contient plus qu'une stack IP.

II Montage de l'image disque Ubuntu

Ceci prend du temps. En effet, c'est comme si on considérait une toute nouvelle machine. Pour cela, il faut utiliser une image disque ubuntu pour virtualbox contrairement à ce que l'on avait fait précédemment. Elle se trouve dans le dossier *rio205*. Le chargement de l'image disque nous propose d'installer ubuntu server, ce que l'on fait. Nous avons créé un compte utilisateur (christophe) avec mot de passe avant de poursuivre. On passe donc à la configuration du disque dur (10 Go), un processeur de 512 Mo de RAM. Une étape importante lors de l'installation est de cocher l'option d'installation d'OpenSSH. En effet, ici, on utilise la connexion de rio205 pour l'installer, ce qui va nous permettre de travailler de la machine virtuelle depuis le terminal rio205.

III Configuration et réseau

On peut ensuite de l'extérieur de la VM accéder aux données. Ici, il n'y a pas de contraintes comme pour le docker car même quand la machine s'éteint, elle sauvegarde son état, notamment les fichiers à l'intérieur. Le logiciel VirtualBox tourne sur un onglet de terminal. Pour paramétrer notre machine virtuelle, il faut d'abord l'éteindre.

Il faut alors effectuer les redirections du port 80 au port 8081 de la localhost et du port 22 au port 2221, dans les paramètres de la machine créée.

Par ailleurs, on met en place l'accélération matérielle. Dans ce cas, c'est le processeur qui redirige lui-même les instructions sensibles pour que la VMM les gère. En effet, sans celle-ci, les solutions de paravirtualisation ou de translation de code exigent de modifier les instructions sensibles du système d'exploitation invité par des sauts à la VMM. Ici, c'est directement les processeurs qui gèrent ce saut vers la VM. Toutes les complications

que l'on pourrait rencontrer avec la virtualisation sont gérées par l'OS hôte qui est aidé par le VMM.

Si l'accès à l'option Accélération dans Système n'est pas activée d'office, il faut aller dans le BIOS de l'ordinateur et activer la virtualisation à ce niveau. Après réallumage de l'ordinateur, on retourne sur rio205 et on relance VirtualBox, on peut alors autoriser dans l'onglet Accélération **l'activation de VT-X /AMD-V**.

Sur la machine invitée, la commande ifconfig nous permet de retrouver l'adresse IP de la VirtualBox (10.0.2.15). Cette adresse correspond à une adresse d'un réseau privé. En effet, dans la configuration, on a la possibilité de choisir l'option NAT, en plus de l'option bridge. C'est ce que l'on a choisi. En ce sens, les machines virtuelles ne voient même pas ce qui est à l'extérieur de leur réseau. Il sont raccordés par l'adresse publique 127.0.0.1. L'attribution de ports de redirection permet le raccord ici.

IV Apache2

On doit pouvoir rentrer dans la machine virtuelle, ce qui se fait par la commande suivante :

```
rio205@rio205:~$ sudo ssh -X -p 2221 christophe@127.0.0.1
```

Ainsi, on peut dans celui-ci, installer ce qui faut pour lancer un serveur apache2. On peut ensuite aller vérifier que l'on peut afficher une page Web (`index.html`). On va donc dans le dossier par défaut qui stocke la page qu'affiche le serveur lorsque l'on ne tape dans un dossier précis du serveur. Après avoir modifié le fichier `index.html`, il faut démarrer le service apache2.

V Analyse et comparaison avec Docker

L'hyperviseur apporté par VirtualBox apporte une sécurité supplémentaire par rapport à la technique précédente. En effet, il filtre toutes les instructions allant jusqu'au hardware, tandis que dans le cas du docker on effectue par défaut toutes les opérations en root. Même sur une machine physique, on n'a pas tous les droits (`sudo`). De ce point de vue, une machine virtuelle apporte plus de sécurité. En effet, les frontières entre le conteneur et la machine ne sont pas aussi bien définies qu'entre une machine invitée et une machine hôte. **En outre, la connexion en NAT semble préférable à la connexion bridge par défaut dans docker, car elle crée un réseau dédié aux machines virtuelles, distinct du réseau de l'hôte.** Enfin, la machine ne sait pas qu'elle fonctionne en mode virtualisé, donc l'accélération matérielle présente des avantages. Les instructions de re-configuration de processeur ou les instructions d'accès aux entrées-sorties peuvent être gérées par le processeur physique lui-même. Néanmoins, cela nécessite certainement des techniques au niveau du hardware pour isoler les ressources pour la VM.

Partie 3

Vagrant

I Lancement de Vagrant

Vagrant est déjà installé. On utilise la documentation fournie par Hashicorp pour avancer. Vagrant prend appui sur VirtualBox pour fonctionner. En effet, ce n'est pas un hyperviseur, mais le logiciel permet d'implémenter des automatisations analogues à docker. Ici, on récupère une image ubuntu pour virtualbox qui va servir de support.

```
rio205@rio205:~$ vagrant init hashicorp/precise64
rio205@rio205:~$ vagrant up
```

Cette commande lance l'image virtualbox et crée par la même occasion un Vagrantfile qui va permettre d'automatiser les étapes de virtualbox.

II Mise à jour du Vagrantfile

Il faut suspendre avec `vagrant suspend` avant de modifier le Vagrantfile. Si on est en connexion ssh (`vagrant ssh`) avec la machine vagrant. Il faut même détruire avec `vagrant destroy` la machine afin de pouvoir relancer la connexion ssh sur la bonne machine. Il suffit d'effectuer `vagrant up` qui va construire à l'aide du Vagrantfile dans le root de rio205.

III Modification du Vagrantfile

Pour cela, il suffit de suspendre la machine et de modifier le Vagrantfile pour scripter la configuration voulue. Il suffit simplement de rajouter quelques lignes pour provisionner vagrant avec des commandes de terminal sur le principe du dockerfile.

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
```

```
Vagrant.configure("2") do |config|
  config.vm.box = "hashicorp/precise64"
```

```
  # accessing "localhost:8080" will access port 80 on the guest machine.
  config.vm.network "forwarded_port", guest: 80, host: 8082, auto_correct: true
  config.vm.network "forwarded_port", guest: 22, host: 2222, auto_correct: true
```

```

config.vm.provision "shell", inline: <<-SHELL
  sudo apt-get update
  sudo apt-get install -y apache2
SHELL
config.vm.provision "file", source: "/htmlpages", destination: "/var/www"
end

```

Ici, on a provisionné vagrant avec le fichier html que nous avons écrit, et nous l'avons placé dans le dossier var/www du roor qui est créé par l'installation d'Apache2. Juste avant, nous avons effectué les redirections.

IV Caractéristiques

On rentre ensuite dans la machine virtuelle par cette commande :

```

rio205@rio205:~$ vagrant ssh
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Welcome to your Vagrant-built virtual machine.
Last login: Wed Mar 27 08:57:56 2019 from 10.0.2.2
vagrant@precise64:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:88:0c:a6
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe88:ca6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:804 errors:0 dropped:0 overruns:0 frame:0
          TX packets:545 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:81394 (81.3 KB)  TX bytes:59676 (59.6 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

```

On voit que l'adresse de sous-réseau de la machine Vagrant n'est pas la même que pour docker par exemple. Il a l'adresse : 10.0.2.15. Néanmoins, l'adresse utile est bien celle de

la localhost. Puis on peut vérifier que le serveur renvoie la page voulue, en lançant depuis rio205 :

```
rio205@rio205:~$ lynx http://127.0.0.1:8080/myindex.html
```

Nous avons remarqué que l'on peut aussi accéder au contenu de l'adresse 10.0.2.15:80 de l'extérieur, c'est-à-dire de rio205 par exemple. En effet, sans configuration supplémentaire dans le Vagrantfile, la connexion n'est ni bridged, ni privée. Il faudrait dans ce cas attribuer directement une adresse privée via le Vagrantfile avec la commande :

```
# Create a private network, which allows host-only access to the machine  
# using a specific IP.  
# config.vm.network "private_network", ip: "192.168.33.10"
```

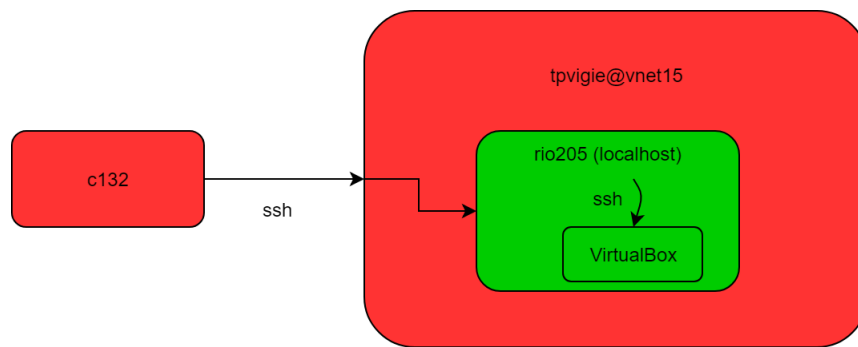
ou bien :

```
# Create a public network, which generally matched to bridged network.  
# Bridged networks make the machine appear as another physical device on  
# your network.  
# config.vm.network "public_network"
```

Ainsi, on peut reproduire les barrières au niveau réseau que l'on avait pour la solution docker.

V Conclusion

La technique Vagrant est utile vis-à-vis de la possibilité d'automatiser à partir d'un fichier texte la création d'une machine virtuelle. Or, dans ce script, il y a le téléchargement d'une image de machine virtuelle entière VirtualBox, ce qui est plus lourd que la création d'image docker. Néanmoins, le site officiel de Vagrant donne des images de machines virtuelles déjà prêtes à être téléchargées. Ainsi, on peut reproduire ce que l'on avait fait avec le dockerfile, notamment par rapport à l'isolation. Or, en terme de sécurité, cette solution présente un atout indéniable par rapport à docker du fait de l'hyperviseur.



Legende :

- Machine physique
- Machine virtuelle

- ➡ Instructions sensibles redirigées vers le VMM
- ➡ Instructions du noyau du système invité

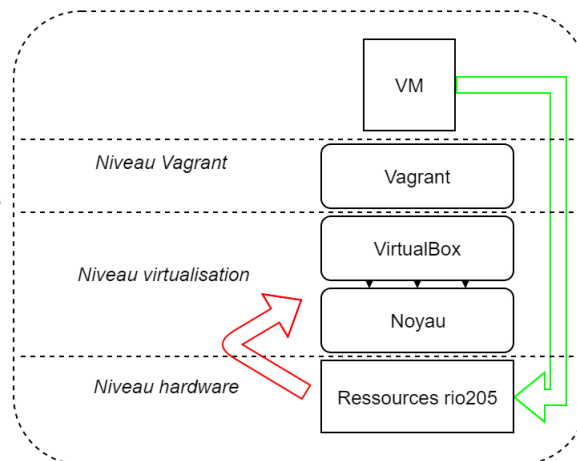


FIGURE 3.1 – Schéma VirtualBox + Vagrant

Partie 4

qemu-kvm

I Préliminaires

On vérifie si le processeur supporte la virtualisation.

```
rio205@rio205:~$ grep -E 'vmx|svm' /proc/cpuinfo
&>/dev/null && echo "oui" || echo "non"
```

Puis on procède ainsi pour que le module noyau soit opérationnel :

```
rio205@rio205:~$ sudo apt-get update
rio205@rio205:~$ sudo apt-get install qemu-kvm libvirt-bin
rio205@rio205:~$ adduser rio205 kvm
rio205@rio205:~$ adduser rio205 libvirt
```

On se déconnecte, reconnecte via ssh pour que les modifications des deux dernières lignes soient prises en compte.

II Création de l'image

```
rio205@rio205:~/VirtualBox VMs/rio205$ VBoxManage clonehd
--format RAW rio205.vdi vmimage.img
rio205@rio205:~/VirtualBox VMs/rio205$ qemu-img convert
-f raw vmimage.img -O qcow2 vmimage.qcow2
```

On installe alors cette image.

III Création de la machine virtuelle

Pour avoir les mêmes propriétés que la machine VirtualBox, on fait ceci.

```
qemu-system-x86_64 vmimage.qcow2 -m 512
-net user,hostfwd=tcp:127.0.0.1:8083-:80,
hostfwd=tcp:127.0.0.1:2223-:22 \
-enable-kvm
```

Puis on accède via ssh au root, et on répète les mêmes étapes que pour les TP précédents.

IV Analyse des configurations possibles

qemu est un support particulièrement intéressant dans le sens où il permet à la fois :

- l'émulation : toutes les instructions inoffensives ou sensibles sont simulées, donc les ressources matérielles sont elles aussi virtuelles (émulateur)
- la virtualisation avec kvm : Seules les instructions sensibles sont simulées (besoin d'un VMM)

En effet, kvm est un module linux qui gère la virtualisation au niveau du noyau linux. **Or, c'est une instance de qemu grâce à la commande enable kvm.** Comme pour VirtualBox, on peut faire une configuration complète du réseau, c'est-à-dire que l'on peut attribuer des adresses privées au VM par exemple.

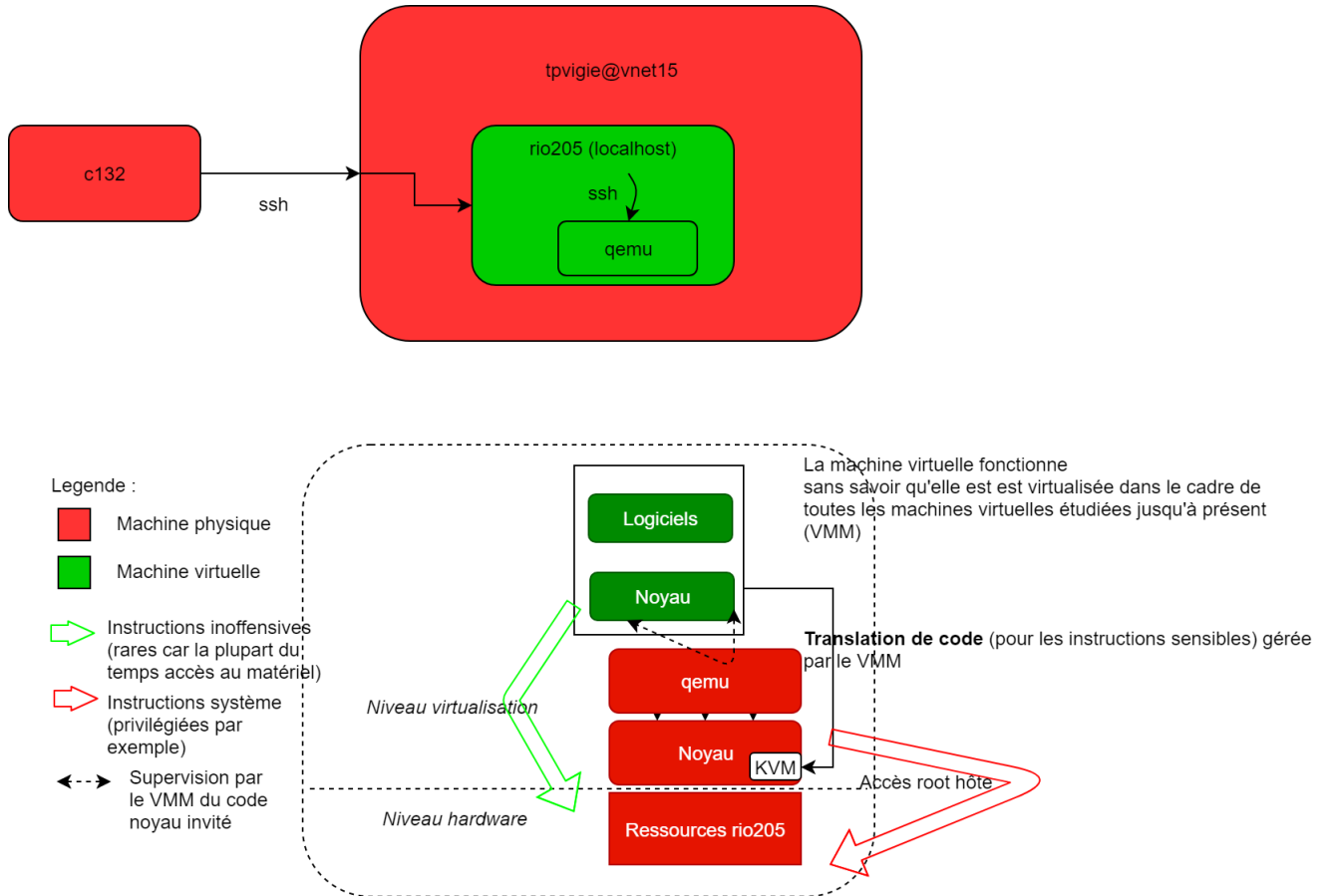


FIGURE 4.1 – Schéma qemu-kvm translation de code

Or, un autre type de virtualisation que l'on a rencontré par exemple dans RIO203 avec Windows quand on a dû utiliser une machine virtuelle ubuntu pour Instant Contiki. C'est la **paravirtualisation**. avec le pilote *virtio* pour KVM. Les performances sont donc accrues. En effet, dans le principe, cette connaissance de la VM des instructions sensibles et de la redirection directe vers le VMM évitent donc à la VM de faire tourner des instructions qu'elle n'est pas capable de gérer rapidement, étant virtualisée. De plus, le VMM voit son travail simplifié.

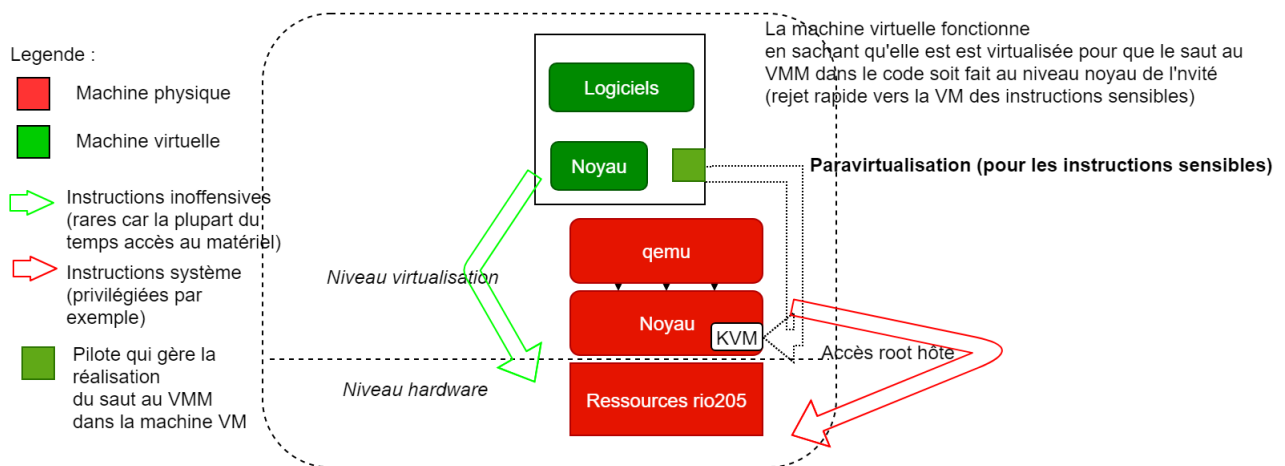


FIGURE 4.2 – Schéma qemu-kvm paravirtualisation

V Bonus

On écrit un script .sh comme lauchScript.sh qui permet d'effectuer plusieurs commandes en une fois dans un shell. On le fait donc dans le langage bash pour créer 10 machines virtuelles. Elles vont différer ici par leur adresse, et donc le port de redirection.

```
#!/bin/bash
for i in `seq 3 13`;
do
    cmd = "qemu-system-x86_64  vmimage.qcow2 -m 512
-net user,hostfwd=tcp:127.0.0.1:808${i}--:80,
hostfwd=tcp:127.0.0.1:222${i}--:22 \
-enable-kvm"
    eval $cmd
done
```

Nous n'avons pas testé ce script, car il faudrait ensuite les tuer, ce qui nécessite aussi un script supplémentaire. De plus, il faudrait s'assurer que l'on puisse utiliser les ports de la plage donnée.

Partie 5

Conclusions et perspectives

Je vois refléter dans mon miroir tout mon passé et tout mon avenir.

J. Cortázar.

I Les leçons apprises

Les différentes techniques de virtualisation nous ont permis de comprendre les enjeux des méthodes de virtualisation :

- la simplicité d'implémentation (docker)
- la consommation en ressources (docker)
- la prise en compte d'une tâche en particulier (docker) ou de fonctionnalités plus complexes (VirtualBox)
- la sécurité du hardware (hyperviseur comme VirtualBox, VMWare)
- les procédures pour lancer et arrêter une solution de virtualisation
- les types de virtualisation (matérielle, translation de code, paravirtualisation)

II Les frontières de la recherche

On pourrait essayer d'autres techniques de virtualisation à hyperviseur qui font de la paravirtualisation par exemple. De même, on pourrait étudier libvirt. Nous aurions aussi pu étudier plus en détail le lancement de plusieurs machines en simultané. Par ailleurs, on aurait pu s'intéresser au déploiement du service Apache2 pour plusieurs fichiers html par exemple ou à une autre tâche plus complexe.

Annexe A

Captures d'écran

Les captures d'écran présentent les étapes par lesquelles nous sommes passés

I Docker

Pour connaître les caractéristiques du réseau :

```
{
  "HostIp": "0.0.0.0",
  "HostPort": "8080"
}
],
"SandboxKey": "/var/run/docker/netns/02ed08b7b6b0",
"SecondaryIPAddresses": null,
"SecondaryIPv6Addresses": null,
"EndpointID": "6fbe9e0fca7c5b181e3b89ffd35faec29841fc962a11
a766531d0",
"Gateway": "172.17.0.1",
"GlobalIPv6Address": "",
"GlobalIPv6PrefixLen": 0,
"IPAddress": "172.17.0.2",
"IPPrefixLen": 16,
"IPv6Gateway": "",
"MacAddress": "02:42:ac:11:00:02",
"Networks": {
  "bridge": {
    "IPAMConfig": null,
    "Links": null,
    "Aliases": null,
```

Nous avons procédé à l'installation manuelle de apache dans le conteneur ainsi :

```
root@c1e1b9b3d548:/# apt-get update
Get:1 http://archive.ubuntu.com/ubuntu bionic InRelease [242 kB]
Get:2 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Get:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:5 http://archive.ubuntu.com/ubuntu bionic/restricted amd64 Packages [13.5
```

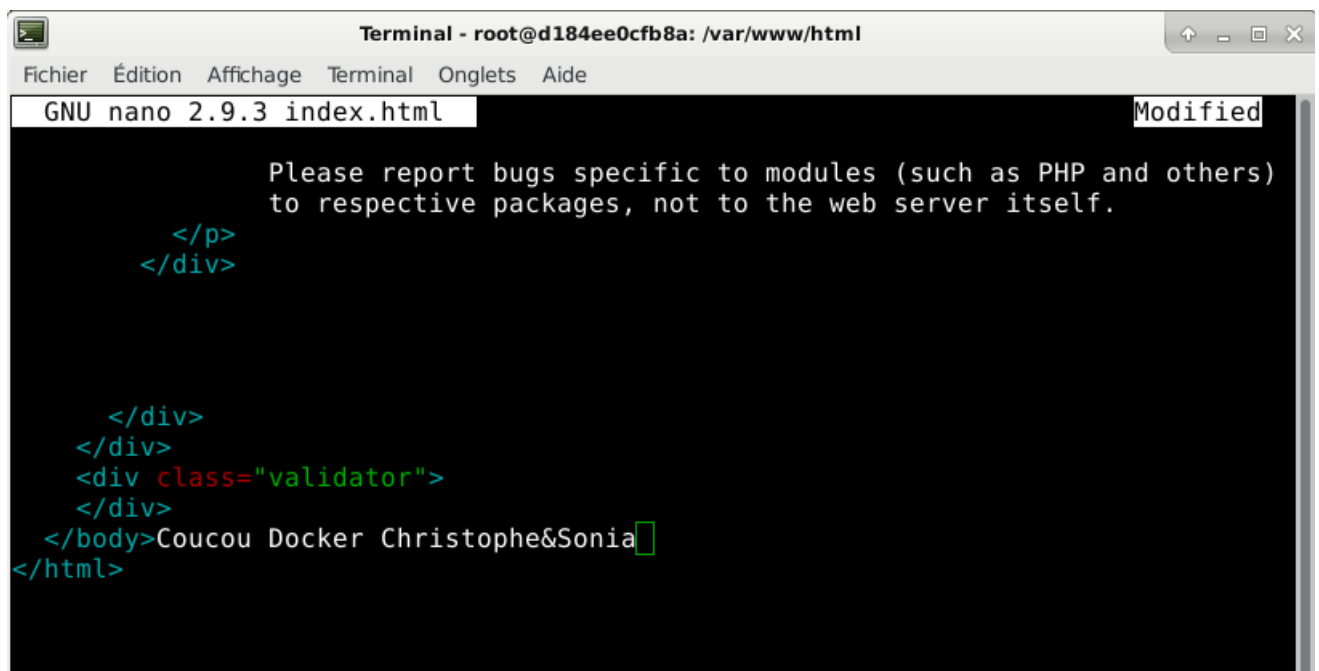
La suite :

On peut aussi utiliser cat si lynx n'est pas installé.

```

root@c1e1b9b3d548:/# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  apache2-bin apache2-data apache2-utils file libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libasn1-8-heimdal libexpat1
  libgdbm-compat4 libgdbm5 libgssapi3-heimdal libhcrypto4-heimdal
  libheimbase1-heimdal libheimntlm0-heimdal libhx509-5-heimdal libicu60
  libkrb5-26-heimdal libldap-2.4-2 libldap-common liblua5.2-0 libmagic-mgc
  libmagic1 libnghttp2-14 libperl5.26 libroken18-heimdal libssl2-2
  libssl2-modules libssl2-modules-db libsqlite3-0 libssl1.1 libwind0-heimdal
  libxml2 mime-support netbase openssl perl perl-modules-5.26 ssl-cert
  xz-utils

```



```

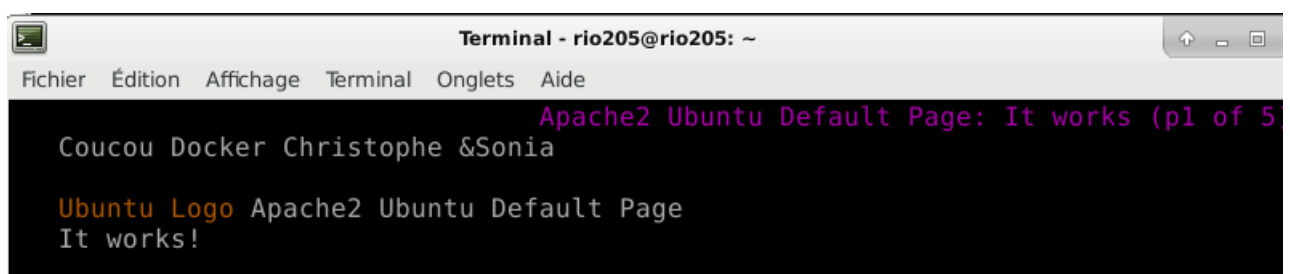
Terminal - root@d184ee0cfb8a: /var/www/html
Fichier  Édition  Affichage  Terminal  Onglets  Aide
GNU nano 2.9.3 index.html Modified

        Please report bugs specific to modules (such as PHP and others)
        to respective packages, not to the web server itself.

    </p>
</div>

</div>
</div>
<div class="validator">
</div>
</body>Coucou Docker Christophe&Sonia
</html>

```



```

Terminal - rio205@rio205: ~
Fichier  Édition  Affichage  Terminal  Onglets  Aide
Apache2 Ubuntu Default Page: It works (p1 of 5)
Coucou Docker Christophe &Sonia
Ubuntu Logo Apache2 Ubuntu Default Page
It works!

```

FIGURE A.1 – Procédure manuelle pour docker

II VirtualBox

Pour voir la configuration réseau de la machine virtuelle invitée :

```

rio205@ubuntu:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:32:2a:d1
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe32:2ad1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:46 errors:0 dropped:0 overruns:0 frame:0
          TX packets:546 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3820 (3.8 KB)  TX bytes:40460 (40.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:11840 (11.8 KB)  TX bytes:11840 (11.8 KB)

```

Pour rentrer dans la machine invitée par ssh :

```

rio205@rio205:~$ sudo ssh -X -p 2221 christophe@127.0.0.1
[sudo] Mot de passe de rio205 :
The authenticity of host '[127.0.0.1]:2221 ([127.0.0.1]:2221)' can't be established.
ECDSA key fingerprint is SHA256:/ooXTPLNyWD6VpMBotedcbthrqBpKM0cBDki5jr1zUQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[127.0.0.1]:2221' (ECDSA) to the list of known hosts
.
christophe@127.0.0.1's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-116-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 paquet peut être mis à jour.
0 mise à jour de sécurité.

Last login: Wed Mar 13 16:43:12 2019
/usr/bin/xdm: file /home/christophe/.Xauthority does not exist

```

Pour éditer le fichier index.html et vérifier la mise en marche du serveur :

```

bug reports</a> before reporting a new bug.
</p>
<p>
Please report bugs specific to modules (such as PHP and others)
to respective packages, not to the web server itself.
</p>
</div>

</div>
</div>
<div class="validator">
</div>
<p>Coucou VirtualBox Sonia & Christophe</p>
</body>
</html>

```

```

rio205@rio205:~$ lynx http://127.0.0.1:8081

```

By default, Ubuntu does not allow access through the web browser to **any** file apart of those located in **/var/www**, **public_html** directories (when enabled) and **/usr/share** (for web applications). If your site is using a web document root located elsewhere (such as in **/srv**) you may need to whitelist your document root directory in **/etc/apache2/apache2.conf**.

The default Ubuntu document root is **/var/www/html**. You can make your own virtual hosts under **/var/www**. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the **ubuntu-bug** tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

Coucou VirtualBox Sonia & Christophe

FIGURE A.2 – Etapes VirtualBox